Boosted Metric Learning for Efficient Identity-Based Face Retrieval
Romain Negrel, Alexis Lechervy, Frédéric Jurie

To cite this version:
Romain Negrel, Alexis Lechervy, Frédéric Jurie. Boosted Metric Learning for Efficient Identity-Based Face Retrieval. 26th British Machine Vision Conference, Sep 2015, Swansea, United Kingdom. 10.5244/C.29.139. hal-01272248

HAL Id: hal-01272248
https://hal.archives-ouvertes.fr/hal-01272248
Submitted on 10 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Boosted Metric Learning for Efficient Identity-Based Face Retrieval

Romain Negrel
romain.negrel@unicaen.fr
Alexis Lechervy
alexis.lechervy@unicaen.fr
Frederic Jurie
frederic.jurie@unicaen.fr
GREYC, CNRS UMR 6072, ENSICAEN
Université de Caen Basse-Normandie
France

Abstract

This paper presents MLBoost, an efficient method for learning to compare face signatures, and shows its application to the hierarchical organization of large face databases. More precisely, the proposed metric learning (ML) algorithm is based on boosting so that the metric is learned iteratively by combining several weak metrics. Boosting allows our method to be free of any hyper-parameters (no cross-validation required) and to be robust with respect to overfitting. This MLBoost algorithm can be trained from constraints involving two pairs of vectors (quadruplets) with a quadratic complexity. The paper also shows how it can be included in a semi-supervised hierarchical clustering framework adapted to identity based face search. Our approach is validated on a benchmark relying on the Labelled Faces in the Wild (LFW) dataset supplemented with 1M face distractors.

1 Introduction and related Work

The performance of many visual recognition algorithms – e.g. face verification or person re-identification – heavily relies on the metric used to measure the similarity between input data. Recent works such as [5, 11, 12, 13, 18] have shown the interest of learning an optimal metric for a particular task using Metric Learning (ML). Most approaches learn a Mahalanobis metric based on an objective function whose constraints come either from a set of labeled examples or, more frequently, from sets of positive (in the same class) and negative (in different classes) pairs.

Presenting the exhaustive list of recent works that have used metric learning (ML) in such a context would be rather tedious, given their large number. However, we can mention a few of the most notable approaches such as DDML [12], RBML [13], Structural ML [11], PCCA [11], rPCCA [11], LMNN [11], LDML [11], ITML [11], KISSME [11], RS-KISSME [11], SML [11]. The reader interested in ML in general is referred to the recent survey by Bellet et al. [11]. The difference among these methods mainly lies in their objective functions which is task dependent or in the type of constraints of the objective function, (e.g. some require pairwise distances between points making large scale applications difficult).

The works of Shen et al. [12] and Bi et al. [11], which have introduced Boosting based approaches, deserve a particular attention due to the interesting properties they offer: i) they
are efficient and scalable, as no semidefinite programming is required and only the largest
eigenvalue and corresponding eigenvectors are needed at each iteration; ii) like AdaBoost,
they don’t have any parameters to tune and they are easy to implement as no sophisticated
optimization techniques are involved. These approaches hence contrast with most of the
commonly used ML methods e.g. [6, 10, 13, 23, 26] for which hyper-parameters, often
introduced for regularization purposes, have to be adjusted by cross-validation.

However, one strong limitation of these two approaches [3, 24] is that they require hav-
ing triplets for learning the metric, i.e., constraints expressed under the form $D(x_a, x_b) <
D(x_a, x_c)$, where $x_a, x_b, x_c$ are three input vectors for which the label is known: positive and
negative pairs within a constraint have to share a common vector. This is a limitation as for
most of the verification task only training pairs are available (e.g., for person re-identification
on the Viper dataset [11] only pairs of same/different persons are provided for training and,
thus, using such triplets is not possible).

One of the key contributions of this paper is to propose a metric learning approach based
on boosting allowing the use of pairs of points for training, i.e. which can use con-
straints such as $D(x_a, x_b) < D(x_c, x_d)$ while keeping the good properties of the approach
in [23] (scalability, simplicity, no parameters, etc.). Our approach is based on the ranking
algorithm RankBoost [8], known to offer three particularly interesting features: i) it requires
no hyperparameters, ii) it has great robustness to overfitting (explained with a standard VC-
dimensional analysis techniques [8, 27]), and iii) it uses a computational trick for reducing
the complexity of the learning step. In the following, we show how to build on RankBoost [8]
to efficiently address this metric learning problem.

The proposed approach is validated on the identity face retrieval task, such as defined
by Bhattarai et al. [10]. This task consists in retrieving the face(s) of the same person to a
query, from a large database of faces. The task is evaluated both in terms of scalability and
accuracy. The idea presented in [10] bears similarity with the work of Verma et al. [22],
who proposed a method for learning hierarchical similarity metrics using a taxonomy. However,
unlike Verma et al., the method proposed by Bhattarai et al. does not require any preexisting
taxonomy. The experimented validation in [22] shows that using the proposed ML algorithm,
allows to improve the performance by a large margin within the same framework as [10].

2 Proposed method : from RankBoost to boosted ML

2.1 Metric learning with boosting

Our approach, referred to as MLBoost, lies in the family of Mahalanobis ML. This means
that the metric $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)$ is parametrized by the positive semi-
definite (PSD) matrix $\mathbf{W}$ and $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}^D \times \mathbb{R}^D$. This metric has two very interesting prop-
erties: (i) it is convex in $\mathbf{W}$ allowing to build convex objective functions; (ii) it can be used
for the dimensionality reduction of the input data. Regarding this last point, we use the fol-
lowing decomposition $\mathbf{W} = \mathbf{L} \mathbf{L}^\top$, with $\mathbf{L} \in \mathbb{R}^{D \times d}$ and $d << D$ the dimension of reduced
space. The data can be projected as: $\mathbf{y}_i = \mathbf{L}^\top \mathbf{x}_i$ and can be compared with the Euclidean
distance: $D_{\mathbf{L}\mathbf{L}^\top}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{L} \mathbf{L}^\top (\mathbf{x}_i - \mathbf{x}_j) = ||\mathbf{y}_i - \mathbf{y}_j||_2^2$.

In the same way boosting builds strong (good) solutions to a problem by the combination of
weak solutions [23], ML with boosting (e.g., [22]) consists in iteratively building a linear
combination of weak Mahalanobis distances of rank 1, denoted by $D_{\mathbf{Z}^{(t)}}(\cdot)$ (with rank($\mathbf{Z}^{(t)}$) = 1
and $\text{tr}(\mathbf{Z}^{(t)}) = 1$): $D_{\mathbf{W}}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^T \alpha^{(t)} D_{\mathbf{Z}^{(t)}}(\mathbf{x}, \mathbf{y})$ in the following.
2.2 Metric Learning with Boosting

We assume that the constraints consist of positive and negative pairs such as \( D_W(p_{1i}, p_{2i}) < D_W(n_{1j}, n_{2j}) \), with \((p_{1i}, p_{2i}) \in P\) the set of similar pairs and \((n_{1j}, n_{2j}) \in N\) the set of dissimilar pairs. This ML problem is closely related the following ranking problem: \( H(p_i) < H(n_j), \forall (p_i, n_i) \), where \( H(\cdot) \) denotes a ranking function and \( p_i = (p_{1i}, p_{2i}) \in P\) and \( n_i = (n_{1j}, n_{2j}) \in N\). We address this problem by building on the RankBoost algorithm [3].

Let us start by associating a weight to each constraint: \( D^{(t)} = \{ d_{ij}^{(t)} \in (0, 1) \mid c_{ij} \in C \} \), with \( C = \{ ((p_{1i}, p_{2i}), (n_{1j}, n_{2j})) \in P \times N \} \) the set of all quadruplets composed by a positive and a negative pair. These weights change over the iterations (denoted as \( t \)) of the algorithm. They can be interpreted as a measure of the importance of the correct ranking the pair \((i, j)\) at the current iteration \( t \). During the iterations, these weights are normalized such that:

\[
\sum_{d_{ij}^{(t)} \in D^{(t)}} d_{ij}^{(t)} = 1. \tag{1}
\]

At the initialization of the algorithm, all quadruples have the same importance and the weights are uniform: \( d_{ij}^{(1)} = \frac{1}{|P||N|}, \forall d_{ij} \in D \). The loss function of MLBoost can be written as:

\[
\text{loss}(W) = \sum_C d_{ij}^{(1)} \left[ D_W(p_{1i}, p_{2i}) \geq D_W(n_{1j}, n_{2j}) \right], \tag{2}
\]

with the function \([\pi]\) that returns \(1\) if predicate \( \pi \) holds and \(0\) otherwise. At each iteration, the weights are updated according to:

\[
d_{ij}^{(t+1)} = \frac{d_{ij}^{(t)} e^{\alpha^{(t)} D_Z^{(t)}(p_{1i}, p_{2i}) - D_Z^{(t)}(n_{1j}, n_{2j})}}{w^{(t)}}, \tag{3}
\]

with \( D_Z^{(t)} \) the weak metric computed at iteration \( t \), \( \alpha^{(t)} \) its weight and \( w^{(t)} \) the normalization factor chosen so that the constraint (1) is met. Equation (3) and the inequality \([x \geq 0] \leq e^x\), allow us to show that:

\[
\text{loss}(W) \leq \prod_{t=1}^{T} w^{(t)}. \tag{4}
\]

The objective at each iteration is then to compute the weak metric \( D_Z^{(t)} \) and the weight \( \alpha^{(t)} \) which minimizes the following function:

\[
w^{(t)} = \sum_C d_{ij}^{(t)} e^{\alpha^{(t)} D_Z^{(t)}(p_{1i}, p_{2i}) - D_Z^{(t)}(n_{1j}, n_{2j})}, \tag{5}
\]

with \( \text{rank}(Z^{(t)}) = 1, \text{tr}(Z^{(t)}) = 1 \) and \( \alpha^{(t)} > 0 \). We note that for increasing the performance of the final metric, we only need to have \( w^{(t)} < 1 \).

**Weak learning algorithm:** The weak metric \( D_Z^{(t)} \) is obtained by computing the solution of an approximation of Equation (5). Freund et al. [3] propose to approximate the exponential by the following lower bound:

\[
e^{\alpha x} \leq \left( \frac{1+x}{2} \right) e^{\alpha} + \left( \frac{1-x}{2} \right) e^{-\alpha}, \tag{6}
\]
for $x \in [-1, 1]$ and $\alpha \in \mathbb{R}$. Using, this idea, obtain the following optimization problem:

$$\arg\min_{\alpha,Z} \left( \frac{1 - r(Z)}{2} e^\alpha + \frac{1 + r(Z)}{2} e^{-\alpha} \right),$$

(7)

where

$$r(Z) = \sum_C d_{ij}^{(t)} \left( D_Z(n_{1j}, n_{2j}) - D_Z(p_{1i}, p_{2i}) \right).$$

(8)

The optimization problem (7) is minimized when $\alpha = \frac{1}{2} \ln \left( \frac{1 + r(Z)}{1 - r(Z)} \right)$, showing that:

$$\prod_{t'=1}^l w^{(t')} \leq \sqrt{1 - r(Z)^2},$$

(9)

Shen et al. \cite{24} proposed to use eq. (8) to compute the weak metric by solving the following optimization problem:

$$Z^{(t)} = \arg\max_Z r(Z)$$

s.t. $\text{rank}(Z) = 1, \text{tr}(Z) = 1$

(10)

By making the following variable change: $Z = zz^\top$, the previous problem is equivalent to:

$$z^{(t)} = \arg\max_z z^\top A^{(t)} z$$

s.t. $\|z\|_2 = 1$

(11)

with $A^{(t)} = \sum_C d_{ij}^{(t)} \left( (n_{1j} - n_{2j})(n_{1j} - n_{2j})^\top - (p_{1i} - p_{2i})(p_{1i} - p_{2i})^\top \right)$. The optimization problem (11) is simply solved by the computation of the eigenvector corresponding to the largest eigenvalue of the $A^{(t)}$ matrix.

Choosing $\alpha^{(t)}$: once the weak metric $D_Z^{(t)}$ is computed with the aforementioned method, the best weight $\alpha^{(t)}$ of this weak metric is computed by solving the following problem:

$$\alpha^{(t)} = \arg\min_\alpha \sum_C d_{ij}^{(t)} e^{\alpha \left( D_Z^{(t)}(p_{1i}, p_{2i}) - D_Z^{(t)}(n_{1j}, n_{2j}) \right)}.$$ 

(12)

This optimization problem is strictly convex and therefore has a unique minimum. We find the solution numerically via the golden section search method. We note that if $\alpha^{(t)}$ is negative or zero, we stop the algorithm. Indeed, we want the parameter matrix of the final metric to meet the positive-semidefinite constraint.

### 2.3 Reduction of the algorithmic complexity

A naive implementation of the presented MLBoost algorithm would induce a high computational complexity. Indeed, the complexity depends on the cardinality of $C$, the set quadruplets. Under the assumption that the images are uniformly distributed across identities, we can approximate the cardinality of $C$ by:

$$|C| = |P||N| \approx \frac{|I|^2}{2c},$$

(13)
Algorithm 1 MLBoost with efficient learning implementation

1: procedure MLBOOST(X, P, N, itersMax)
2:    \( t \leftarrow 1 \)
3:    \( \mathbf{L}^{(t)} \leftarrow \emptyset \)
4:    Initialize weights: \( u_i^{(t)} = 1/|\mathcal{P}|, \forall i \in 1 \ldots |\mathcal{P}|, v_j^{(t)} = 1/|\mathcal{N}|, \forall j \in 1 \ldots |\mathcal{N}|. \)
5:    repeat
6:        Compute weak metric: \( \mathbf{z}^{(t)} \) by solving of eq. (11) + (16).
7:        Choose the best: \( \alpha^{(t)} \) by solving of eq. (17).
8:        if \( \alpha^{(t)} \leq 0 \) then
9:            break
10:        \( \mathbf{L}^{(t+1)} \leftarrow [\mathbf{L}^{(t)}, \sqrt{\alpha^{(t)}} \mathbf{z}^{(t)}] \)
11:    Update weights: \( u_i^{(t+1)} \) and \( v_j^{(t)} \) with the eq. (15).
12: until \( t < \text{itersMax} \)
13: return \( \mathbf{L} \)

with \(|\mathcal{I}|\) the total number of images and \(c\) the number of identities. The computational complexity, as a function of the number of images, is thus \( \mathcal{O}(|\mathcal{I}|^4) \).

Interestingly, Freund et al. [\textsc{[9]}] have introduced a way to rewrite the equations of Rank-Boost in order to reduce the computational complexity. It turns out these principles extend to our MLBoost algorithm. For doing this, we decompose \( d_{ij}^{(t)} \), the weights of quadruplets in \( \mathcal{C} \), as follows:

\[
d_{ij}^{(t)} = u_i^{(t)} v_j^{(t)}, \quad (14)
\]

with \( u_i^{(1)} = 1/|\mathcal{P}|, \forall i \) and \( v_j^{(1)} = 1/|\mathcal{N}|, \forall j \). Equation (3) used to update the weights can be rewritten as:

\[
u_j^{(t+1)} = \frac{u_i^{(t)} e^{\alpha^{(t)} D_{ij}(p_i, p_j)}}{w_P^{(t)}}, \forall i
\]

\[
u_j^{(t+1)} = \frac{v_j^{(t)} e^{-\alpha^{(t)} D_{ij}(n_1, n_2)}}{w_N^{(t)}}, \forall j
\]

with \( w_P^{(t)} \) and \( w_N^{(t)} \) the normalization factors chosen such that \( \sum u_i^{(t+1)} = 1 \) and \( \sum v_j^{(t+1)} = 1 \).

The expression of matrix \( \mathbf{A}^{(t)} \) used to compute the weak metric (11) is rewritten as follows:

\[
A^{(t)} = \sum_{\forall j} v_j^{(t)} \left( (n_{1j} - n_{2j})(n_{1j} - n_{2j})^\top \right) - \sum_{\forall p} u_i^{(t)} \left( (p_{1i} - p_{2i})(p_{1i} - p_{2i})^\top \right). \quad (16)
\]

The optimization problem (12) allowing to compute \( \alpha^{(t)} \), which are the weights of this weak metric, is rewritten as:

\[
\alpha^{(t)} = \arg \min_{\alpha} \left( \sum_{\forall p} u_i^{(t)} e^{\alpha D_{ij}(p_i, p_j)} \right) \left( \sum_{\forall j} v_j^{(t)} e^{-\alpha D_{ij}(n_1, n_2)} \right). \quad (17)
\]

In this form, the equations have a much lower computational complexity which it depends on the sum of the number of positive and negative pairs : \(|\mathcal{P}| + |\mathcal{N}| \). The computational complexity is then of the order of \( \mathcal{O}(|\mathcal{I}|^2) \). The different steps of MLBoost are described in Algorithm 1.
Algorithm 2

1: procedure HIERARCHICAL_TREE_CLUSTERING(X, y, nbPairs, nbCluster, treeDepth)
2:   rootNode.idxC ← [1...N] and rootNode.idxML ← [1...N]
3:   WorkQueue.push(rootNode)
4:   while WorkQueue.empty() == false do
5:     node ← WorkQueue.pop()
6:     (P, N) ← build_pairs(y[node.idxML], nbPairs)
7:     L ← RankBoost(X[:, node.idxML], P, N, 1024)
8:     node.L ← L
9:     if node.depth < treeDepth then
10:        X' ← L†X[:, node.idxC]
11:        C ← clustering(X', nbCluster)
12:        for i = 1...nbCluster do
13:           childNode.idxC ← cluster_assign(node.idxC, X', C, i)
14:           childNode.idxML ← ml_selection(y, childNode.idxC)
15:           node.addChild(childNode)
16:           WorkQueue.push(childNode)
17:   return rootNode

3 MLBoost for Semi-Supervised Hierarchical Clustering

In [2], Bhattarai et al. proposed a method for building hierarchical face representations for efficient identity-based face retrieval. The main idea is to use a ML algorithm for building a tree structured representation of a large set of faces, such that identical-identity faces belong to the same cluster at each tree level. The hierarchical clustering is built recursively: at each level, a new metric is learned using the faces belonging to a particular tree node which is then used to locally cluster the faces and to create the sub-branches.

The retrieval for a new query face is done by visiting the tree from top to bottom until a leaf is reached; if the current node is a non-leaf node, the face is projected into the corresponding subspace and compared with its centroids, which allows to move to the closest child node. When a leaf is reached, the face is compared with all the faces contained in the leaf using the corresponding local metric.

The main drawback of this method is that, when a local metric is learned within a particular node of the tree, no information is retained from the parent nodes. By doing this, the metrics become quickly too local (over specialized). Furthermore, misclassified faces (those going in to the wrong branches) can strongly impact the local metrics.

We therefore propose to improve [2] by combining the metrics learnt at different levels of the tree. A simple method for doing this would be to add an elastic term between the metric from the parent branch and the metric learned. However, the addition of these terms would add hyper-parameters to the metric learning problem and would require some additional tuning. We propose instead a simple method based on the selection of training samples: if one or more examples of an identity belongs to a given node, we use all the examples of this identity to learn the metric as described in Algo. 2. We see that, in each branch, we have two vectors of example indices: the one of the faces actually belonging to the branch (denoted as idxC in Algo. 2 this vector is the same as that used in the algorithm proposed by [3]); the example indices used to learn the metric in the branch, denoted as idxML in Algo. 2.
4 Experiments

Performance evaluation. In the considered scenario: a person has to be identified given an image of his face by querying a dataset containing faces of different identities. The performance should reflect the ability of the system to place the correct identity at the beginning of a list of candidate identities. The criterion used to evaluate the performance is the one used in [2], i.e., the mean of $k$-call@n (such as defined in [6]), with $k = 1$. We measure the performance for several values of $n$, i.e., $n \in \{1, 10, 20, 50, 100\}$.

Database: We use the well known Labeled Faces in the Wild (LFW) database by Huang et al. [16]. This database has more than 13000 images of over 4000 different identity. We use its aligned version with deep funneling [15]. In our experiments, we use the same set of images and queries than the ones used by Bhattarai et al. [2]. We keep only the images of identities with at least five examples each, all other images are not used during the experiments; it results in a set of 5985 images of 423 persons. Regarding the query set, we select one query image per identity (the same as Bhattarai et al.). The training set contains the selected images, excluding the query set. To evaluate the robustness with respect to large-scale databases, we use the set of million distractor faces kindly provided by Bhattarai et al. [2].

Image description: We evaluate the methods with two types of image signatures: (i) Local Binary Pattern (LBP) [20] and (ii) Fisher Vector (FV) [21, 22]. To extract the LBP signature, we use the same protocol as proposed by Bhattarai et al., each image being described by a vector of 9860 dimensions [2, 28]. To extract the FV signature, we use the implementation proposed by Simonyan et al. [25]. We use the publicly available source code with the default parameters and with our own training set. Each image is then described by a vector of 67584 dimensions. The two types of signatures (LBP and FV) are subsequently $\ell_2$-normalized.

Training pairs: We select as many positive pairs as negative ones, uniformly distributed across the identities.

Dimension of the output space: The proposed algorithm does not allow to control explicitly the dimension of the output space, which is required for fair comparisons (i.e., for comparing two approaches for a given dimensionality of the projected features). Thus, we select the $d$ best projectors by applying a singular value decomposition of the projectors resulting in a $d$-dimensional signature.

Baseline (PCA, no metric learning): These experiments are intended to evaluate the performance obtained by the raw signatures (without metric learning), possibly projected in a low dimensional space by PCA. They are reported in Table 1, which compares the performance of the two signatures (LBP and FV), without distractors or with 100.000 and 1M distractors, and for different values of recall $n$. The performances are given in the mean 1-call@n (%). Without distractors, the FV signatures perform better than LBP, while with distractors LBP outperforms FV. Projecting the signatures on a subspace of 4096 dimensions or more does not alter the performance significantly. In the following experiments, we pre-reduce the signatures at 4096 components with the PCA projectors.

Experimental validation of the proposed algorithm (exhaustive search)

Figure 1(a) illustrates some results on convergence of the algorithm (without distractors, using $2 \times 32768$ pairs for training). Interestingly, we do not see any overfitting (this is confirmed by adding even more iterations). Figure 1(b) shows the performance of the algorithm
Table 1: Mean 1-call@\(n\) performance given by raw LBP and FV signatures (no ML), with/without PCA, with/without distractors.

<table>
<thead>
<tr>
<th>Sign.</th>
<th>Dim.</th>
<th>No distractor</th>
<th>+100,000 distractors</th>
<th>+1,000,000 distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(n=1) 10 20 50 100</td>
<td>(n=1) 10 20 50 100</td>
<td>(n=1) 10 20 50 100</td>
</tr>
<tr>
<td>LBP</td>
<td>9860</td>
<td>31.9 53.7 60.5 68.8 74.7</td>
<td>31.4 52.7 59.3 67.8 73.3</td>
<td>31 50.6 57.2 65.0 70.4</td>
</tr>
<tr>
<td>LBP-PCA</td>
<td>8192</td>
<td>31.9 53.7 60.5 68.8 74.7</td>
<td>31.4 52.2 58.4 66.7 72.1</td>
<td>30.3 49.4 55.8 61.9 69.0</td>
</tr>
<tr>
<td>LBP-PCA</td>
<td>4096</td>
<td>31.7 52.7 60.3 68.1 74.2</td>
<td>30.5 49.6 56.5 62.6 69.3</td>
<td>28.1 44.7 48.9 57.2 62.4</td>
</tr>
<tr>
<td>FV-PCA</td>
<td>32</td>
<td>16.1 33.3 43.3 55.8 66.7</td>
<td>15.1 31.4 39.5 51.3 60.3</td>
<td>13.2 29.1 33.3 42.3 50.1</td>
</tr>
</tbody>
</table>

Table 2: Comparison with PCCA [2].
Hierarchical tree structure for efficient large-scale identity-based face retrieval

This last part of this experimental section focuses on the use of our MLBoost algorithm for face retrieval, in the context of semi-supervised hierarchical clustering, with/without distractors. Here again, we use LBP and FV signatures and set the number of training pairs to $2 \times 16384$ and the dimension of the output space to 32. We restrict ourselves to a binary clustering tree of 3 and 4 levels (denoted with suffix “-d3” and “-d4”). Table 3 reports the 1-recall@n performance for the two descriptors, for different depths of the tree and for different amounts of distractors.

<table>
<thead>
<tr>
<th>Sign.</th>
<th>Method</th>
<th>Dim.</th>
<th>n=1</th>
<th>n=10</th>
<th>n=20</th>
<th>n=1</th>
<th>n=10</th>
<th>n=20</th>
<th>n=1</th>
<th>n=10</th>
<th>n=20</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP</td>
<td>MLBoost</td>
<td>32</td>
<td>30.7</td>
<td>53.7</td>
<td>62.2</td>
<td>26.2</td>
<td>43.0</td>
<td>48.7</td>
<td>21.5</td>
<td>34.5</td>
<td>38.5</td>
</tr>
<tr>
<td>LBP</td>
<td>MLBoost-Tree-d3-Global</td>
<td>32</td>
<td>27.9</td>
<td>50.4</td>
<td>58.6</td>
<td>24.1</td>
<td>39.5</td>
<td>46.1</td>
<td>20.1</td>
<td>31.4</td>
<td>36.2</td>
</tr>
<tr>
<td>LBP</td>
<td>MLBoost-Tree-d3</td>
<td>32</td>
<td>29.6</td>
<td>54.6</td>
<td>62.2</td>
<td>23.6</td>
<td>39.5</td>
<td>45.2</td>
<td>17.7</td>
<td>31.0</td>
<td>35.2</td>
</tr>
<tr>
<td>LBP</td>
<td>PCCA-Tree-d3 [1]</td>
<td>32</td>
<td>-</td>
<td>33.3</td>
<td>40.9</td>
<td>-</td>
<td>27.4</td>
<td>32.2</td>
<td>-</td>
<td>18.5</td>
<td>23.9</td>
</tr>
<tr>
<td>LBP</td>
<td>MLBoost-Tree-d4-Global</td>
<td>32</td>
<td>27.0</td>
<td>49.2</td>
<td>56.7</td>
<td>22.7</td>
<td>37.4</td>
<td>42.3</td>
<td>18.9</td>
<td>30.5</td>
<td>33.6</td>
</tr>
<tr>
<td>LBP</td>
<td>MLBoost-Tree-d4</td>
<td>32</td>
<td>28.6</td>
<td>52.0</td>
<td>59.3</td>
<td>21.0</td>
<td>35.5</td>
<td>40.9</td>
<td>17.3</td>
<td>26.7</td>
<td>30.5</td>
</tr>
<tr>
<td>LBP</td>
<td>PCCA-Tree-d4 [1]</td>
<td>32</td>
<td>-</td>
<td>36.2</td>
<td>41.6</td>
<td>-</td>
<td>32.4</td>
<td>36.6</td>
<td>-</td>
<td>30.3</td>
<td>31.7</td>
</tr>
<tr>
<td>FV</td>
<td>MLBoost</td>
<td>32</td>
<td>44.7</td>
<td>69.0</td>
<td>78.0</td>
<td>28.6</td>
<td>48.0</td>
<td>52.5</td>
<td>20.1</td>
<td>34.0</td>
<td>38.3</td>
</tr>
<tr>
<td>FV</td>
<td>MLBoost-Tree-d3-Global</td>
<td>32</td>
<td>42.6</td>
<td>65.7</td>
<td>75.7</td>
<td>27.7</td>
<td>45.9</td>
<td>48.9</td>
<td>19.1</td>
<td>33.3</td>
<td>37.1</td>
</tr>
<tr>
<td>FV</td>
<td>MLBoost-Tree-d3</td>
<td>32</td>
<td>48.0</td>
<td>69.0</td>
<td>75.2</td>
<td>30.3</td>
<td>42.8</td>
<td>45.4</td>
<td>22.0</td>
<td>33.1</td>
<td>36.4</td>
</tr>
<tr>
<td>FV</td>
<td>MLBoost-Tree-d4-Global</td>
<td>32</td>
<td>43.0</td>
<td>66.0</td>
<td>75.2</td>
<td>27.6</td>
<td>45.4</td>
<td>48.9</td>
<td>19.1</td>
<td>33.3</td>
<td>36.9</td>
</tr>
<tr>
<td>FV</td>
<td>MLBoost-Tree-d4</td>
<td>32</td>
<td>51.1</td>
<td>70.2</td>
<td>73.8</td>
<td>27.0</td>
<td>39.2</td>
<td>42.3</td>
<td>22.5</td>
<td>30.3</td>
<td>31.7</td>
</tr>
</tbody>
</table>

Table 3: Performance of the hierarchical tree structure and comparison with PCCA [1].

Searching the faces in only one leaf of the tree affects the overall performance: misclassifications that can happen at any level of the tree cannot be recovered later. This loss increases with the depth of the tree. We can observe this by comparing the performance of the global metric used without and with the tree structure (“MLBoost” in Table 3) (methods whose names include “MLBoost-Tree-d*-Global”). The loss is of about 1-2% at $n = 1$ with FV.

We can also observe that the local metric (the lines denoted as “MLBoost-Tree-d*”) improves the performance of the tree-based structure for the small values of $n$, not only compensating entirely the loss induced by the tree but also giving a significant improvement of more than 6% ($n = 1$, without distractors, with FV). Comparing our method with the state-of-the-art method of [1] (lines denoted as “PCCA-Tree-d*”) shows that our methods can perform up to 30% better for some configurations. We can also see that the FV signature provides significant gain compared to the LBP signature.

5 Conclusions

This paper has introduced a boosted ML method called MLBoost that, in contrast with most of ML methods, does not require any hyper-parameters to be tuned and is not subject to overfitting. The paper also shows how to use the proposed MLBoost algorithm for computing hierarchical organizations of face databases allowing to reduce the computational cost of identity-based face retrieval. The proposed experimental validation has demonstrated the robustness of the method w.r.t. overfitting. In addition, its performance is significantly better than PCCA, which is one of the state-of-the-art ML method. Finally, local ML has been shown to provide better performance than global ML in this context.
Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-12-SECU-0005 (project PHYSIONOMIE).

References


